Creating and using date/timestamp partitioned tables

This document describes how to create and use tables partitioned by a `DATE` or `TIMESTAMP` column. For information on ingestion-time partitioned tables, see Creating and using ingestion-time partitioned tables (/bigquery/docs/creating-partitioned-tables). For information on integer range partitioned tables, see Creating and using integer range partitioned tables (/bigquery/docs/creating-integer-range-partitions).

After creating a partitioned table, you can:

- Control access to your table data
- Get information about your partitioned tables
- List the partitioned tables in a dataset
- Get partitioned table metadata using meta-tables

For more information on managing partitioned tables including updating partitioned table properties, copying a partitioned table, and deleting a partitioned table, see Managing partitioned tables (/bigquery/docs/managing-partitioned-tables).

Limitations

Partitioned tables are subject to the following limitations:

- The partitioning column must be either a scalar `DATE` or `TIMESTAMP` column. While the mode of the column may be `REQUIRED` or `NULLABLE`, it cannot be `REPEATED` (array-based).
- The partitioning column must be a top-level field. You cannot use a leaf field from a `RECORD` (`STRUCT`) as the partitioning column.
- You cannot use legacy SQL to query partitioned tables or to write query results to partitioned tables.

Hourly partitioned tables are currently subject to further limitations:

- Hourly partitioned tables cannot be manipulated in the web UI.

- The `bq partition` command in the `bq` command-line tool is unsupported.

Creating partitioned tables

You can create a partitioned table in BigQuery by:

- Using the Cloud Console or the classic web UI
- Using a DDL `CREATE TABLE` statement with a `PARTITION BY` clause containing a `partition expression` (/bigquery/docs/data-definition-language#partition_expression)
- By using the command line tool's `bq mk` command
- Programmatically by calling the `tables.insert` (/bigquery/docs/reference/rest/v2/tables/insert) API method
- From query results (/bigquery/docs/writing-results)
- When you load data (/bigquery/docs/loading-data)

Table naming

When you create a table in BigQuery, the table name must be unique per dataset. The table name can:

- Contain up to 1,024 characters
- Contain letters (upper or lower case), numbers, and underscores

Required permissions

At a minimum, to create a table, you must be granted the following permissions:

- `bigquery.tables.create` permissions to create the table
- `bigquery.tables.updateData` to write data to the table by using a load job, a query job, or a copy job
- `bigquery.jobs.create` to run a query job, load job, or copy job that writes data to the table

Additional permissions such as `bigquery.tables.getData` might be required to access the data you're writing to the table.

The following predefined IAM roles include both `bigquery.tables.create` and `bigquery.tables.updateData` permissions:

- `bigquery.dataEditor`
- `bigquery.dataOwner`
- `bigquery.admin`

The following predefined IAM roles include `bigquery.jobs.create` permissions:

- `bigquery.user`
- `bigquery.jobUser`
- `bigquery.admin`

In addition, if a user has `bigquery.datasets.create` permissions, when that user creates a dataset, they are granted `bigquery.dataOwner` access to it. `bigquery.dataOwner` access gives the user the ability to create and update tables in the dataset.

For more information on IAM roles and permissions in BigQuery, see Predefined roles and permissions (/bigquery/docs/access-control).

Daily partitioning vs hourly partitioning

When using a `TIMESTAMP` column to partition data, you can create partitions with either hourly or daily granularity, depending on your data and needs.

Daily partitioning is the default partitioning type and, when used with clustering, serves the majority of BigQuery use cases. In particular, daily partitioning is the better choice when your data is spread out over a wide range of dates, or if data is continuously added over time. If your data spans a wide range of dates, daily partitioning allows you to remain under your table's partition limits (/bigquery/quotas#partitioned_tables).

Choose hourly partitioning instead if your tables have a high volume of data that spans a short date range (typically less than six months of timestamp values). With hourly partitioning, you can address data at hour-level granularity; for example, when appending, truncating, or deleting data from a particular partition.

Creating an empty partitioned table with a schema definition

You cannot create an empty partitioned table that does not have a schema definition. The schema is required in order to identify the column used to create the partitions.

When you create an empty partitioned table with a schema definition, you can:

- Provide the schema inline using the `bq` command-line tool
- Specify a JSON schema file using the `bq` command-line tool
- Provide the schema in a table resource (/bigquery/docs/reference/rest/v2/tables) when calling the API's `tables.insert` (/bigquery/docs/reference/rest/v2/tables/insert) method

For more information on specifying a table schema, see Specifying a schema (/bigquery/docs/schemas).

After the partitioned table is created, you can:

- Load data into it
- Write query results to it
- Copy data into it

To create an empty partitioned table with a schema definition:

ConsoleDDL (#ddl)Classic UI (#classic-ui)bq (#bq)API (#api)Go (#go)Java (#java)Node.js (#node.js)Py

1. In the navigation panel, in the **Resources** section, expand your project and select your dataset.
2. On the right side of the window, in the details panel, click **Create table**.
3. In the **Create table** panel, in the **Source** section:
    - For **Create table from**, select **Empty table**.
4. In the **Destination** section:
    - For **Dataset name**, choose the appropriate dataset, and in the **Table name** field, enter the name of the table you're creating.
    - Verify that **Table type** is set to **Native table**.
5. In the **Schema** section, enter the schema (/bigquery/docs/schemas) definition.
    - Enter schema information manually by:

- Enabling **Edit as text** and entering the table schema as a JSON array.

> ★ **Note:** You can view the schema of an existing table in JSON format by entering the following command in the **bq** command-line tool: **`bq show --format=prettyjson`** ***`dataset.table`***.

- Using **Add field** to manually input the schema.

6. For **Partition and cluster settings**, click **No partition**, select **Partition by field** and choose the `DATE` or `TIMESTAMP` column. This option is not available if the schema does not contain a `DATE` or `TIMESTAMP` column.
7. (Optional) For **Partitioning filter**, click the **Require partition filter** box to require users to include a `WHERE` clause that specifies the partitions to query. Requiring a partition filter may reduce cost and improve performance. For more information, see Querying partitioned tables (/bigquery/docs/querying-partitioned-tables#querying_partitioned_tables_2).
8. (Optional) Click **Advanced options** and for **Encryption**, click **Customer-managed key** to use a Cloud Key Management Service key (/bigquery/docs/customer-managed-encryption). If you leave the **Google-managed key** setting, BigQuery encrypts the data at rest (/security/encryption-at-rest/default-encryption).
9. Click **Create table**.

Note: When you create an empty partitioned table by using the Cloud Console, you cannot add a label, description, table expiration, or partition expiration. You can add these optional properties when you create a table by using the API or **bq** command-line tool.

After the table is created, you can update the partitioned table's table expiration (/bigquery/docs/managing-tables#updating_a_tables_expiration_time), description (/bigquery/docs/managing-partitioned-tables#updating_a_partitioned_tables_description), and labels (/bigquery/docs/adding-labels). You cannot use the Cloud Console to add a partition expiration after a table is created.

Creating partitioned tables from query results

To create a partitioned table from a query result, write the results to a new destination table. You can create a partitioned table by querying either a partitioned table or a non-partitioned table. You cannot change an existing standard table to a partitioned table using query results.

When you create a partitioned table from a query result, you must use standard SQL. Currently, legacy SQL is not supported for querying partitioned tables or for writing query results to

partitioned tables.

Partition decorators enable you to write the query results to a specific partition. For example, to write the results to the May 1, 2016 partition, use the following partition decorator:

```
_name$20160501
```

When writing query results to a specific partition using a partition decorator, the data that is being written to the partition must conform to the table's partitioning scheme. All rows written to the partition should have values that fall within the partition's date.

For example:

The following query retrieves data from February 1, 2018 and writes the data to the `$20180201` partition of table `mytable`. The table has two columns — a `TIMESTAMP` column named `TS` and an `INT64` column named `a`.

```
ery \
se_legacy_sql  \
tination_table=mytable$20180201 \
CT
MESTAMP("2018-02-01") AS TS,
AS a'
```

The following query retrieves data from January 31, 2018, and attempts to write the data to the `$20180201` partition of `mytable`. This query fails because the data you're attempting to write doesn't fall within the partition's date.

```
ery \
se_legacy_sql  \
tination_table=T$20180201 \
CT
MESTAMP("2018-01-31") as TS,
as a'
```

For information on appending to or restating (replacing) data in partitioned tables, see Appending to and overwriting time partitioned table data

(/bigquery/docs/managing-partitioned-table-data#append-overwrite). For more information on querying partitioned tables, see Querying partitioned tables (/bigquery/docs/querying-partitioned-tables).

Creating a partitioned table from a query result

To create a partitioned table from a query result:

Console Classic UI (#classic-ui) bq (#bq) API (#api)

You cannot specify partitioning options for a destination table when you query data using the Cloud Console.

Creating a partitioned table when loading data

You can create a partitioned table by specifying partitioning options when you load data into a new table. You do not need to create an empty partitioned table before loading data into it. You can create the partitioned table and load your data at the same time.

When you load data into BigQuery, you can supply the table schema, or for supported data formats, you can use schema auto-detection (/bigquery/docs/schema-detect).

Partition decorators enable you to load data into a specific partition. For example, to load all data that is generated on May 1, 2016 into the `20160501` partition, use the following partition decorator:

```
_name$20160501
```

When loading data into a specific partition using a partition decorator, the data that is being loaded into the partition must conform to the table's partitioning scheme. All rows written to the partition should have values that fall within the partition's date.

For more information on loading data, see Introduction to loading data into BigQuery (/bigquery/docs/loading-data).

Hourly partitioning with clustering

Hourly partitioning can be used with clustering. Clustering an hourly partitioned table will first partition its data by the hour boundaries of the partitioning table, then cluster it further by the clustering columns.

As an example, this command creates a table with an hourly partitioned column and a cluster.

```
q mk --time_partitioning_type=HOUR \
-time_partitioning_field=ts_column \
-clustering_fields=ts_column,column1 \
ydataset.mytable2 "ts_column:TIMESTAMP,column1:INTEGER,column2:STRING"
```

When you retrieve the format of the table, you will see that both hourly timestamp partitioning and clustering are in effect:

```
q show --format=prettyjson mydataset.mytable2
..
 "clustering": {
   "fields": [
     "ts_column",
     "column1"
   ]
 },
..
 "timePartitioning": {
   "field": "ts_column",
   "type": "HOUR"
 },
..
```

Controlling access to partitioned tables

To configure access to tables and views, you can grant an IAM role to an entity at the following levels, listed in order of range of resources allowed (largest to smallest):

- a high level in the Google Cloud resource hierarchy (/resource-manager/docs/cloud-platform-resource-hierarchy) such as the project, folder, or organization level
- the dataset level
- the table/view level

Access with any resource protected by IAM is additive. For example, if an entity does not have access at the high level such as a project, you could grant the entity access at the dataset level, and then the entity will have access to the tables and views in the dataset. Similarly, if the entity does not have access at the high level or the dataset level, you could grant the entity access at the table of view level.

Granting IAM roles at a higher level in the Google Cloud resource hierarchy (/resource-manager/docs/cloud-platform-resource-hierarchy) such as the project, folder, or organization level gives the entity access to a broad set of resources. For example, granting a role to an entity at the project level gives that entity permissions that apply to all datasets throughout the project.

Granting a role at the dataset level specifies the operations an entity is allowed to perform on tables and views in that specific dataset, even if the entity does not have access at a higher level. For information on configuring dataset-level access controls, see Controlling access to datasets (/bigquery/docs/dataset-access-controls).

Granting a role at the table or view level specifies the operations an entity is allowed to perform on specific tables and views, even if the entity does not have access at a higher level. For information on configuring table-level access controls, see Controlling access to tables and views (/bigquery/docs/table-access-controls).

You can also create IAM custom roles (/iam/docs/creating-custom-roles). If you create a custom role, the permissions you grant depend on the specific operations you want the entity to be able to perform.

You can't set a "deny" permission on any resource protected by IAM.

For more information on roles and permissions, see:

- Understanding roles (/iam/docs/understanding-roles) in the IAM documentation
- BigQuery Predefined roles and permissions (/bigquery/docs/access-control)

- Controlling access to datasets (/bigquery/docs/dataset-access-controls)
- Controlling access to tables and views (/bigquery/docs/table-access-controls)
- Restricting access with BigQuery Column-level security (/bigquery/docs/column-level-security)

Using partitioned tables

Getting information about partitioned tables

You can get information about tables by:

- Using the Cloud Console or the classic BigQuery web UI
- Using the `bq show` command in the `bq` command-line tool
- Calling the `tables.get` (/bigquery/docs/reference/rest/v2/tables/get) API method
- Using the client libraries

Required permissions

At a minimum, to get information about tables, you must be granted `bigquery.tables.get` permissions. The following predefined IAM roles include `bigquery.tables.get` permissions:

- `bigquery.metadataViewer`
- `bigquery.dataViewer`
- `bigquery.dataOwner`
- `bigquery.dataEditor`
- `bigquery.admin`

In addition, if a user has `bigquery.datasets.create` permissions, when that user creates a dataset, they are granted `bigquery.dataOwner` access to it. `bigquery.dataOwner` access gives the user the ability to retrieve table metadata.

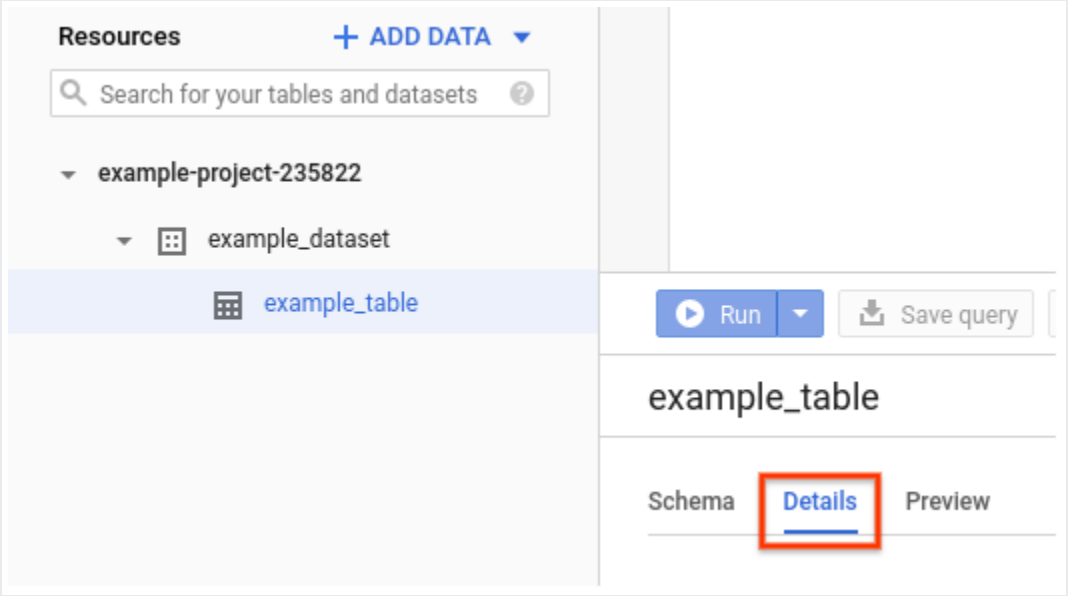
For more information on IAM roles and permissions in BigQuery, see Access control (/bigquery/access-control).

Getting partitioned table information

To view information about a partitioned table:

Console Classic UI (#classic-ui) bq (#bq) API (#api)

1. Open the BigQuery web UI in the Cloud Console.
   Go to the BigQuery web UI (https://console.cloud.google.com/bigquery)
2. In the navigation panel, in the **Resources** section, expand your project and dataset, then click the table name in the list.
3. Click **Details** below the **Query editor**. This tab displays the table's description and table information.



4. Click the **Schema** tab to view the table's schema definition. Notice partitioned tables do not include the `_PARTITIONTIME` pseudo column.

Listing partitioned tables in a dataset

You can list tables in datasets (including partitioned tables) by:

- Using the Cloud Console or the classic BigQuery web UI
- Using the `bq ls` command in the `bq` command-line tool
- Calling the `tables.list` (/bigquery/docs/reference/rest/v2/tables/list) API method

- Using the client libraries

Required permissions

At a minimum, to list tables in a dataset, you must be granted `bigquery.tables.list` permissions. The following predefined IAM roles include `bigquery.tables.list` permissions:

- `bigquery.user`
- `bigquery.metadataViewer`
- `bigquery.dataViewer`
- `bigquery.dataEditor`
- `bigquery.dataOwner`
- `bigquery.admin`

For more information on IAM roles and permissions in BigQuery, see Access control (/bigquery/access-control).

Listing partitioned tables

To list the tables in a dataset (including partitioned tables):

Console Classic UI (#classic-ui) bq (#bq) API (#api)

1. Open the BigQuery web UI in the Cloud Console.
   Go to the BigQuery web UI (https://console.cloud.google.com/bigquery)
2. In the navigation panel, in the **Resources** section, expand your project and click on your dataset.
3. Scroll through the list to see the tables in the dataset. Tables, partitioned tables, models, and views are identified by different icons.

Listing partitions in partitioned tables

You can list the partitions in a partitioned table by querying the `__PARTITIONS_SUMMARY__` meta table using legacy SQL.

You can run the query in the Cloud Console, the classic BigQuery web UI, using the `bq query` command, or by calling the `jobs.insert` (/bigquery/docs/reference/rest/v2/jobs/insert) method and configuring a `query` (/bigquery/docs/reference/v2/jobs#configuration.query) job.

Required permissions

At a minimum, to run a query job that uses the `__PARTITIONS_SUMMARY__` meta-table, you must be granted `bigquery.jobs.create` permissions. The following predefined IAM roles include `bigquery.jobs.create` permissions:

- `bigquery.user`
- `bigquery.jobUser`
- `bigquery.admin`

You must also be granted `bigquery.tables.getData` permissions. The following predefined IAM roles include `bigquery.tables.getData` permissions:

- `bigquery.dataViewer`
- `bigquery.dataEditor`
- `bigquery.dataOwner`
- `bigquery.admin`

For more information on IAM roles in BigQuery, see Access control (/bigquery/access-control).

Listing partitions in a partitioned table

You can list partitions in a partitioned table using legacy SQL. To list partitions in a partitioned table:

Console Classic UI (#classic-ui) bq (#bq) API (#api)

1. Open the BigQuery web UI in the Cloud Console.
   Go to the Cloud Console (https://console.cloud.google.com/bigquery)
2. Click the **Compose new query** button.

3. Enter the following text into the **Query editor** box to query the `__PARTITIONS_SUMMARY__` meta-table:

```
#legacySQL
SELECT
  partition_id
FROM
  [dataset.table$__PARTITIONS_SUMMARY__]
```

   Where:

   - ***dataset*** is the dataset that contains the table.
   - ***table*** is the name of the table.

4. Click **Run**.

Getting partitioned table metadata using meta tables

You can get information about partitioned tables by using special tables called meta tables. Meta tables contain metadata such as the list of tables and views in a dataset. The meta tables are read only.

Currently, you cannot use the `INFORMATION_SCHEMA` service to get partitioned table metadata.

Getting partition metadata using meta tables

The `__PARTITIONS_SUMMARY__` meta table is a special table whose contents represent metadata about partitions in a time-partitioned table. The `__PARTITIONS_SUMMARY__` meta table is read-only.

To access metadata about the partitions in a time-partitioned table, use the `__PARTITIONS_SUMMARY__` meta-table in a query's `SELECT` statement. You can run the query by:

- Using the Cloud Console or the classic BigQuery web UI
- Using the command-line tool's `bq query` command
- Calling the `jobs.insert` (/bigquery/docs/reference/rest/v2/jobs/insert) API method and configuring a `query` job

- Using the client libraries

Currently, standard SQL does not support the partition decorator separator ($) so you cannot query `__PARTITIONS_SUMMARY__` in standard SQL. A legacy SQL query that uses the `__PARTITIONS_SUMMARY__` meta-table looks like the following:

```
T
umn

taset.table$__PARTITIONS_SUMMARY__]
```

Where:

- ***dataset*** is the name of your dataset.
- ***table*** is the name of the time-partitioned table.
- ***column*** is one of the following:

Value	Description
`project_id`	Name of the project.
`dataset_id`	Name of the dataset.
`table_id`	Name of the time-partitioned table.
`partition_id`	Name (date) of the partition.
`creation_time`	The time at which the partition was created, in milliseconds since January 1, 1970 UTC.
`last_modified_time`	The time at which the partition was last modified, in milliseconds since January 1, 1970 UTC.

Partition meta table permissions

At a minimum, to run a query job that uses the `__PARTITIONS_SUMMARY__` meta-table, you must be granted `bigquery.jobs.create` permissions. The following predefined IAM roles include `bigquery.jobs.create` permissions:

- `bigquery.user`
- `bigquery.jobUser`
- `bigquery.admin`

You must also be granted `bigquery.tables.getData` permissions. The following predefined IAM roles include `bigquery.tables.getData` permissions:

- `bigquery.dataViewer`
- `bigquery.dataEditor`
- `bigquery.dataOwner`
- `bigquery.admin`

For more information on IAM roles in BigQuery, see Access control (/bigquery/access-control).

Partition meta table examples

The following query retrieves all partition metadata for a time-partitioned table named `mydataset.mytable`.

Console Classic UI (#classic-ui) bq (#bq)

```
#legacySQL
SELECT
  *
FROM
  [mydataset.mytable$__PARTITIONS_SUMMARY__]
```

The output looks like the following:

```
------------+------------+----------------+--------------+---------------+----------
roject_id   | dataset_id |    table_id    | partition_id | creation_time | last_modi
------------+------------+----------------+--------------+---------------+----------
roject      | mydataset  | mytable        | 20160314     | 1517190224120 | 151719022
roject      | mydataset  | mytable        | 20160315     | 1517190224120 | 151719022
------------+------------+----------------+--------------+---------------+----------
```

The following query lists the times when the partitions in `mydataset.mytable` were last modified.

Console Classic UI (#classic-ui) bq (#bq)

```
#legacySQL
SELECT
  partition_id,
  last_modified_time
FROM
  [mydataset.mytable$__PARTITIONS_SUMMARY__]
```

The output looks like the following:

```
----------+--------------------+
tition_id | last_modified_time |
----------+--------------------+
60102     |      1471632556179 |
60101     |      1471632538142 |
60103     |      1471632570463 |
----------+--------------------+
```

To display the `last_modified_time` field in human-readable format, use the `FORMAT_UTC_USEC` function. For example:

Console Classic UI (#classic-ui) bq (#bq)

```
#legacySQL
SELECT
  partition_id,
  FORMAT_UTC_USEC(last_modified_time*1000) AS last_modified
FROM
  [mydataset.table1$__PARTITIONS_SUMMARY__]
```

The output looks like the following:

```
----------+----------------------------+
tition_id |       last_modified        |
----------+----------------------------+
60103     | 2016-08-19 18:49:30.463000 |
60102     | 2016-08-19 18:49:16.179000 |
60101     | 2016-08-19 18:48:58.142000 |
----------+----------------------------+
```

Next steps

- For an overview of partitioned table support in BigQuery, see Introduction to partitioned tables (/bigquery/docs/partitioned-tables).
- To learn how to create and use ingestion-time partitioned tables, see Creating and using ingestion-time partitioned tables (/bigquery/docs/creating-partitioned-tables).
- To learn how to create and use integer range partitioned tables, see Creating and using integer range partitioned tables (/bigquery/docs/creating-integer-range-partitions).
- To learn how to manage and update partitioned tables, see Managing partitioned tables (/bigquery/docs/managing-partitioned-tables).
- For information on querying partitioned tables, see Querying partitioned tables (/bigquery/docs/querying-partitioned-tables).

Last updated 2020-08-19 UTC.